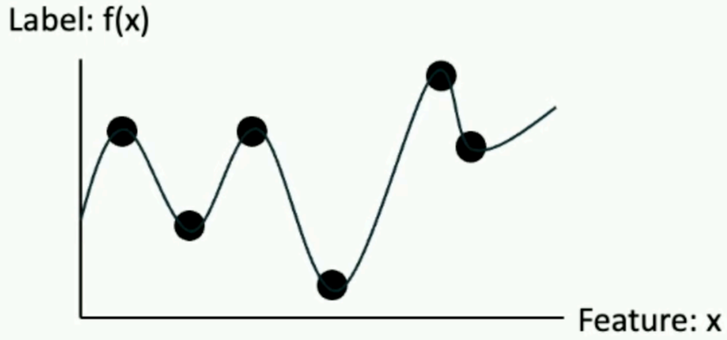
Average Gradient Outer Product as a Mechanism for Feature Learning

fzeng 2025-09-19

Outline

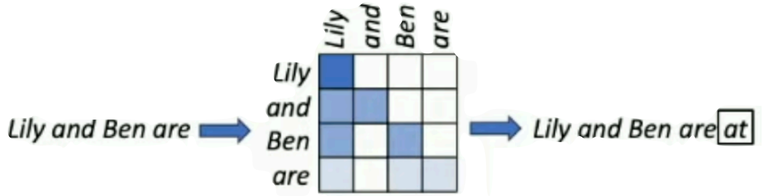
- Feature learning
- Deep Neural Feature Ansatz (NFA)
- Empirical evidence for NFA
- Theoretical results for NFA
- NFA and notable deep learning phenomena
- Recursive feature machines & applications to LLM concept detection/steering
- Other recent work
- Takeaways

The curse of dimensionality, and feature learning



Number of features	Number of samples
1	n
2	n^2
d	n^d

Transformer based large language models (LLMs)
(e.g., GPT4, Claude)

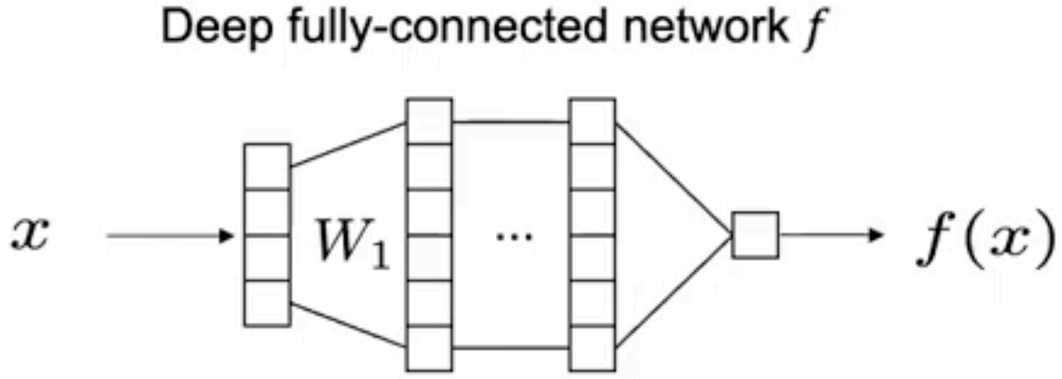


Trained using:
> 100k context size
+
> 100k vocab size (# of tokens)

Neural networks overcome the curse of dimensionality via feature learning

Neural Feature Matrix

- Claim: the gram matrix $M := W_i^T W_i$ featurizes inputs
- SVD: $W = USV^T$
- Extracts features with rotation/scaling: SV^T
- Applies a rotation/reflection: U
- Learned features: $SV^T x = (W^T W)^{1/2} x$



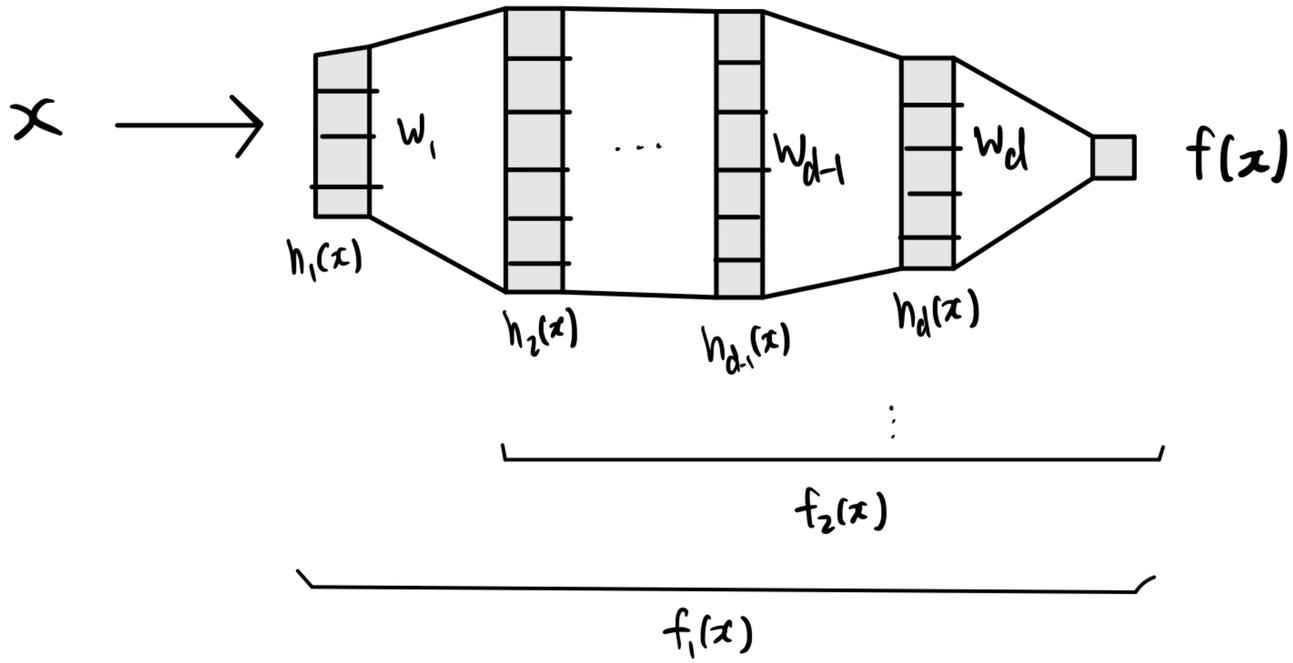
- So feature learning is learning an inner product defined by M: $\langle M^{1/2}x, M^{1/2}x \rangle = x^T M z$
- We call M the **Neural Feature Matrix**

Average gradient outer product (AGOP)

- Jacobian $\nabla f(x_i)$: tells us the directions in input space that are most influential in computing our output
- For a predictor f on point x, $\text{AGOP}(f, X) = \sum_i \nabla f(x_i) \nabla f(x_i)^T$
- (Uncentered) covariance matrix of sensitivity to input
- Why does this seem helpful?
  - Features that are relevant should influence output
  - So AGOP recovers the relevant features

Deep Neural Feature Ansatz (NFA)

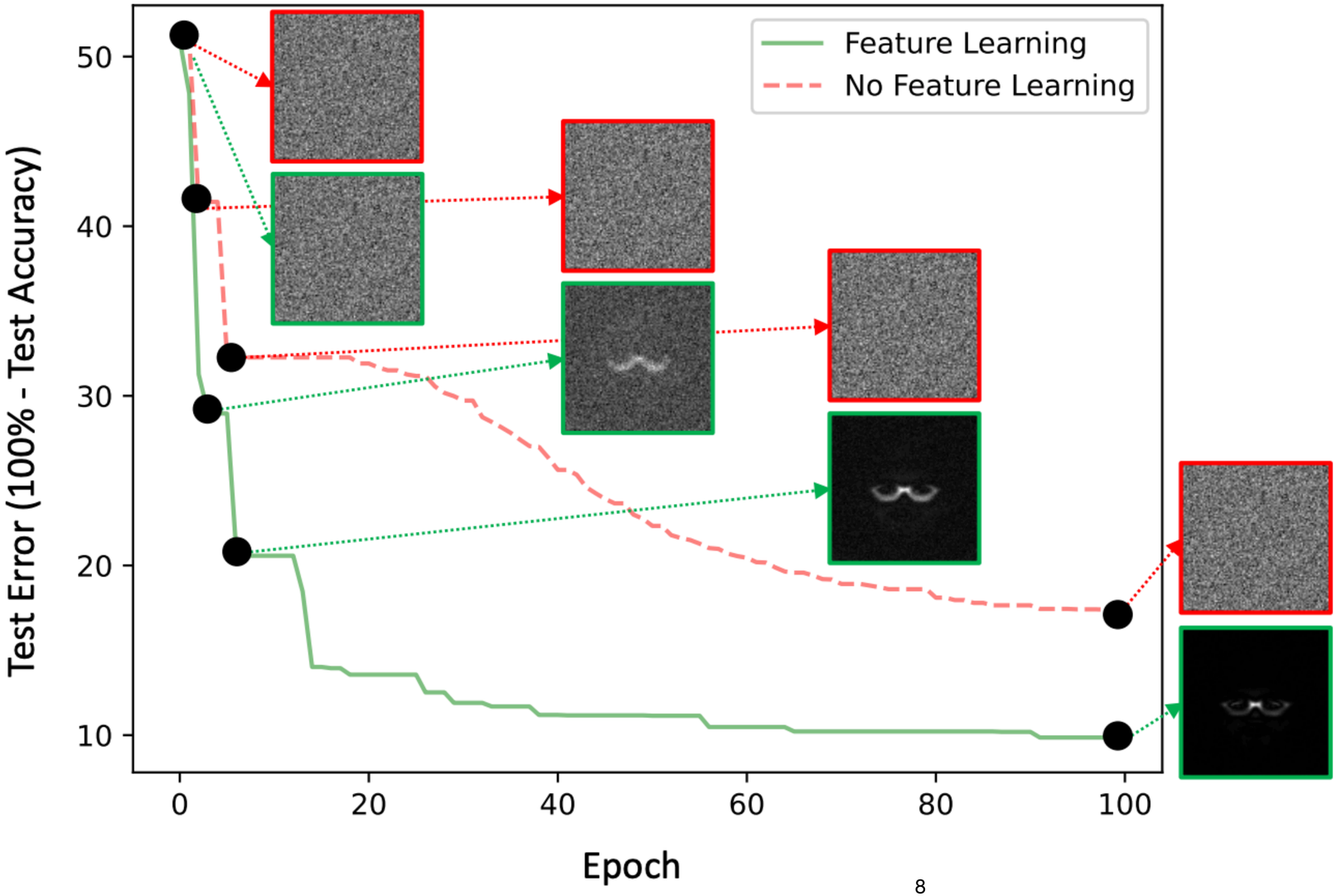
- For a deep nonlinear fully connected NN f, denoting weights on layer i by W_i and inputs for layer i by $h_i(x)$, and the subnetwork that operates on $h_i(x)$ as f_i
- Then throughout training, for each layer i, the Neural Feature Matrix $W_i^T W_i$ is proportional to Average Gradient Outer Product of f_i with respect to $h_i(x)$:



$$W_i^T W_i \propto \frac{1}{n} \sum_{p=1}^{n} \nabla f_i(h_i(x_p)) \nabla f_i(h_i(x_p))^T$$

Empirical evidence for NFA

A Neural feature learning when classifying glasses in images



2 layer NN

Red: first layer frozen

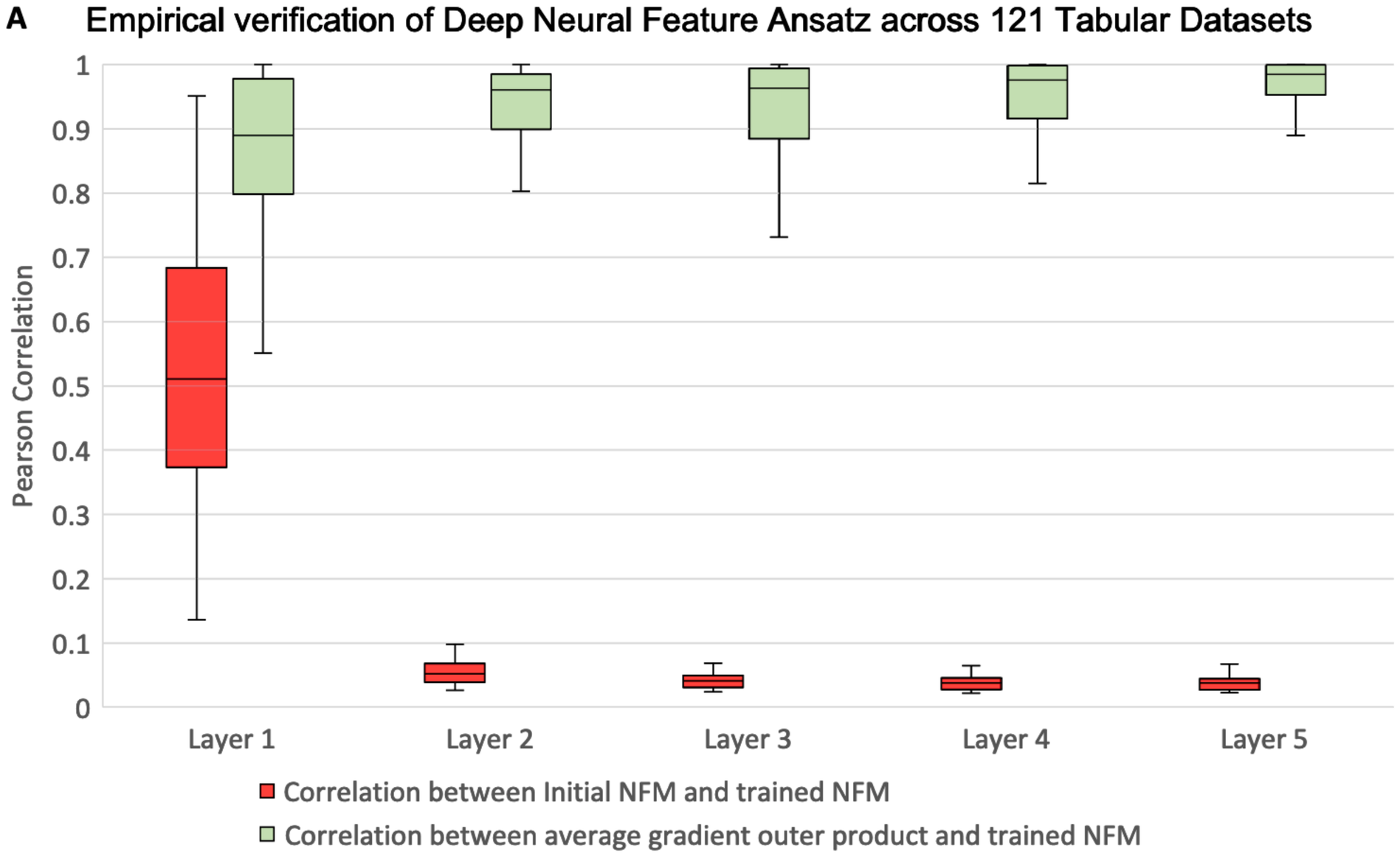
Green: all layers trained

Diagonal of first layer NFM visualized

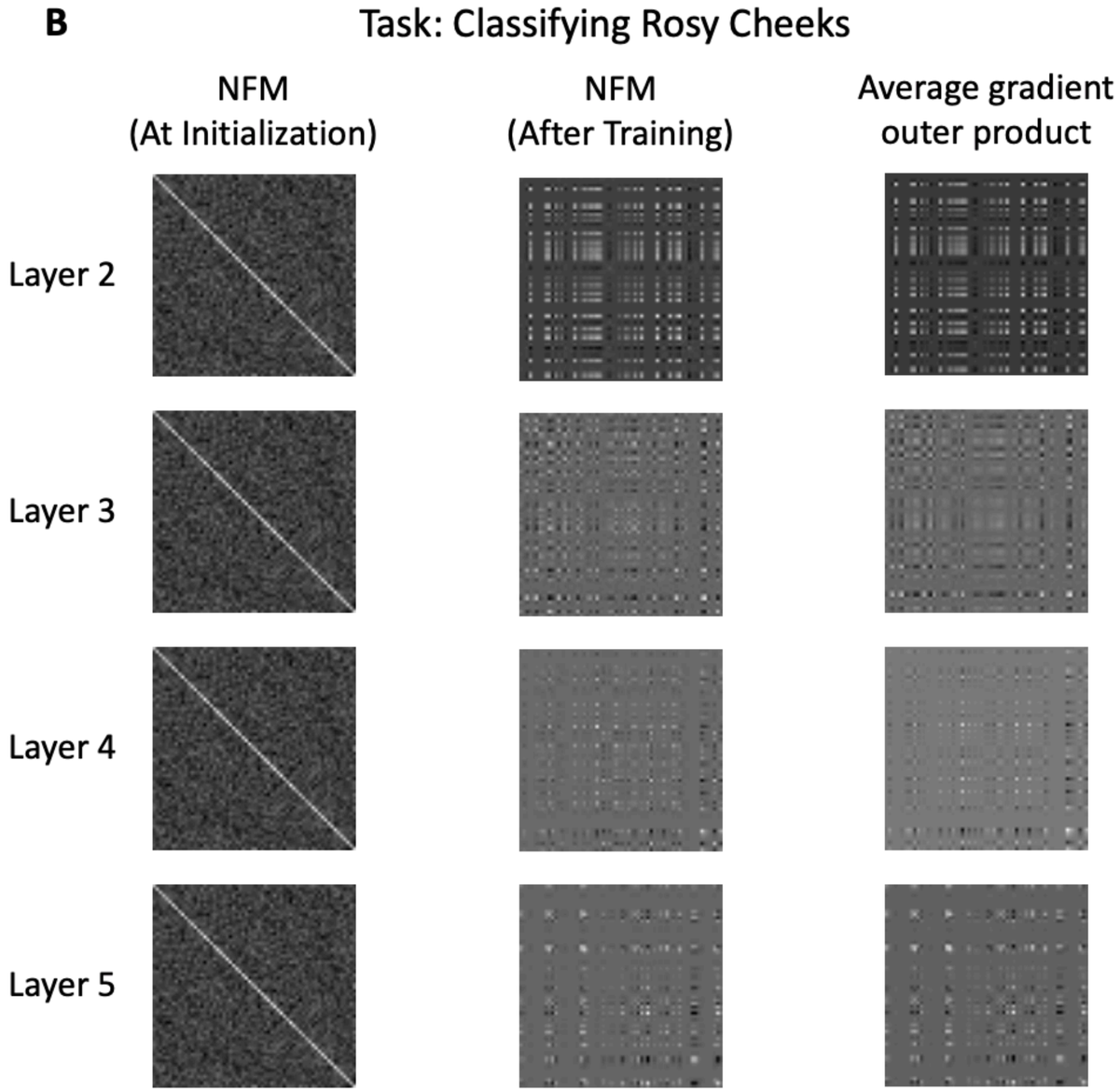
B Average gradient outer product captures neural network features

Classification Task	5 o'clock shadow	Necktie	Smiling	Rosy Cheeks
Correlation between first layer NFM and Avg. gradient outer product	0.993	0.970	0.998	0.991
NFM of first layer (Top Eigenvector)				
Avg. gradient outer product (Top Eigenvector)				

NFM and AGOP visually indistinguishable



Lower corr for first layer due to input dim being smaller



NFM: ~identity at initialization

Converges to AGOP after training (corr >.78)

Theoretical results for NFA

Simple case

- Consider a NN where first layer is fully connected and are the only trainable weights
- Weight matrix 0 initialized
- One datapoint, multiple steps of GD
- MSE loss
- Let’s prove it!

Proposition 1. *Let $f(z) = g(Bz)$ with $f : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^k \to \mathbb{R}$. Given one training sample (x, y), suppose that f is trained to minimize $\frac{1}{2}(y - f(x))^2$ using gradient descent. Let $B^{(\ell)}$ denote B after ℓ steps of gradient descent. If $B^{(0)} = \mathbf{0}$ and $f_t(z) := g(B^{(t)}z)$, then for all time steps t:*

$$\nabla f_t(z) \nabla f_t(z)^T \propto {B^{(t)}}^T B^{(t)} \ .$$

Proof. Gradient descent with learning rate η proceeds as follows:

$$B^{(t+1)} = B^{(t)} + \eta \nabla g(B^{(t)}x)(y - g(B^{(t)}x))x^T \ .$$

If $B^{(0)} = \mathbf{0}$, then by induction $B^{(t)} = \alpha^{(t)} x^T$ for all time steps t where $\alpha^{(t)} \in \mathbb{R}^k$. Then, we have that

$$\begin{aligned}
\nabla f_t(z) \nabla f_t(z)^T &= {B^{(t)}}^T \nabla g(B^{(t)}z) g(B^{(t)}z)^T B^{(t)} \\
&= x {\alpha^{(t)}}^T \nabla g(B^{(t)}z) g(B^{(t)}z)^T \alpha^{(t)} x^T \\
&= (xx^T)({\alpha^{(t)}}^T \nabla g(B^{(t)}z) g(B^{(t)}z)^T \alpha^{(t)}) \\
&\propto xx^T \ .
\end{aligned}$$

Similarly, we have that

$${B^{(t)}}^T B^{(t)} = x {\alpha^{(t)}}^T \alpha^{(t)} x^T = (xx^T)({\alpha^{(t)}}^T \alpha^{(t)}) \propto xx^T \ .$$

Other settings NFA is proven to hold

Result	Activation	Steps	Depth	Outer layers	Initialization	GIA	# Samples
Proposition 1	Any	Any	Any	Fixed	Zero	No	1
Proposition 2	Any	1	Any	Fixed	Zero	No	Any
Proposition 3	Linear	Any	2	Fixed, i.i.d.	Zero	No	Any
Proposition 4	Linear	2	2	Trainable, i.i.d	Zero	No	Any
Theorem 1	ReLU	Any	Any	Fixed, i.i.d.	Any	Yes	Any

First layer fully connected & trainable

Other layers either fixed or trainable

Initialization: for first layer weights

Gradient Independence Assumption: replace weights in backprop by weighs from the same distribution to simplify analysis

Deep Neural Feature Ansatz sheds light on notable phenomena from deep learning

Spurious Features

Spurious Features

(A) **Cow: 0.99**, Pasture: 0.99, Grass: 0.99, No Person: 0.98, Mammal: 0.98

(B) No Person: 0.99, Water: 0.98, Beach: 0.97, Outdoors: 0.97, Seashore: 0.97

(C) No Person: 0.97, **Mammal: 0.96**, Water: 0.94, Beach: 0.94, Two: 0.94

Neural networks often rely on features that are likely to co-occur with the target feature but not part of it, i.e “spurious features”

Spurious Features

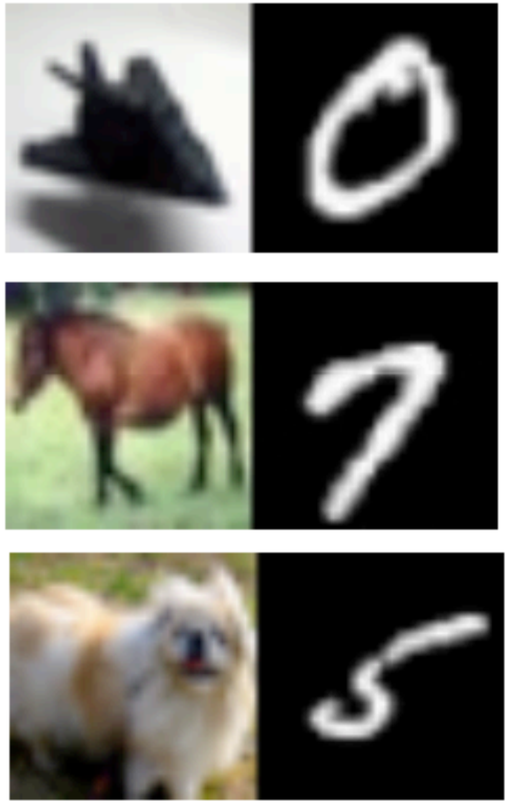
(a) **class:** band aid, **spurious feature**: fingers, **-41.54%** (b) **class:** space bar, **spurious feature**: keys, **-46.15%** (c) **class:** plate, **spurious feature**: food, **-32.31%** (d) **class:** butterfly, **spurious feature**: flowers, **-21.54%** (e) **class:** potter's wheel, **spurious feature**: vase, **-21.54%**

Second row: amplified features (via gradient ascent), like asking for "more" of the label

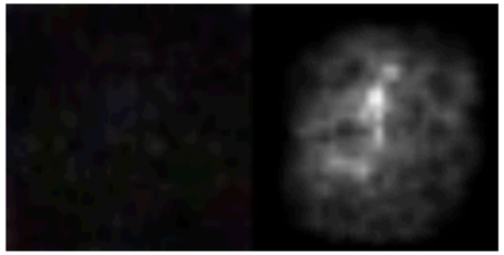
Red: % change in classification accuracy if Gaussian noise added to spurious regions

Concatenated CIFAR10 and MNIST

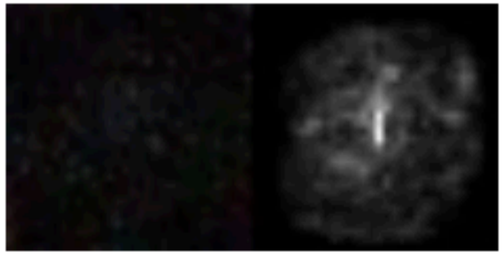
Samples from
Combined CIFAR10-MNIST



NFM
(Diagonal)



Avg. gradient outer product
(Diagonal)



Trucks and planes with colored star

Samples from Modified STL-10

NFM (Diagonal)

Avg. gradient outer product (Diagonal)

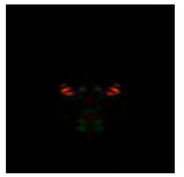
CelebA lipstick classification

Lipstick Classification

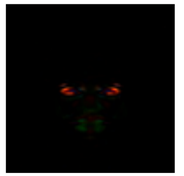
40k training, 20k test samples from CelebA

MLP Test Accuracy: 90.79%

NFM
(Diagonal)



Avg. gradient outer product
(Diagonal)

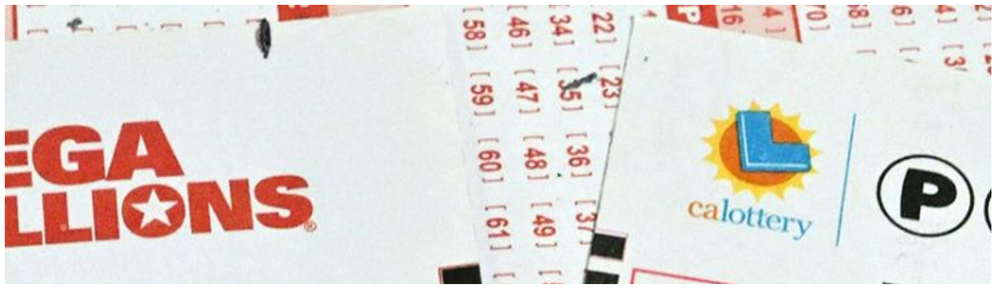


Corruption	Mask	Test Acc.
None		90.79%
Lips (1260 Pixels)		87.13%
Eyes (477 Pixels)		63.57%

Takeaway: NFA can be used to identify spurious features

Lottery Tickets

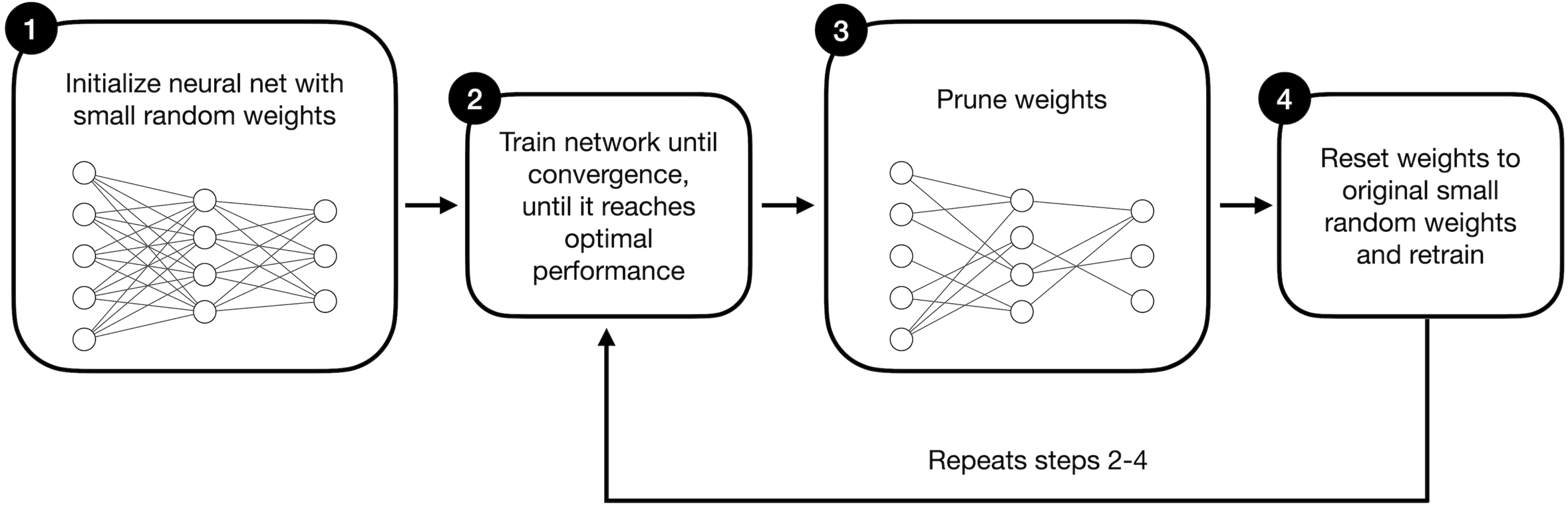
Lottery Ticket Hypothesis



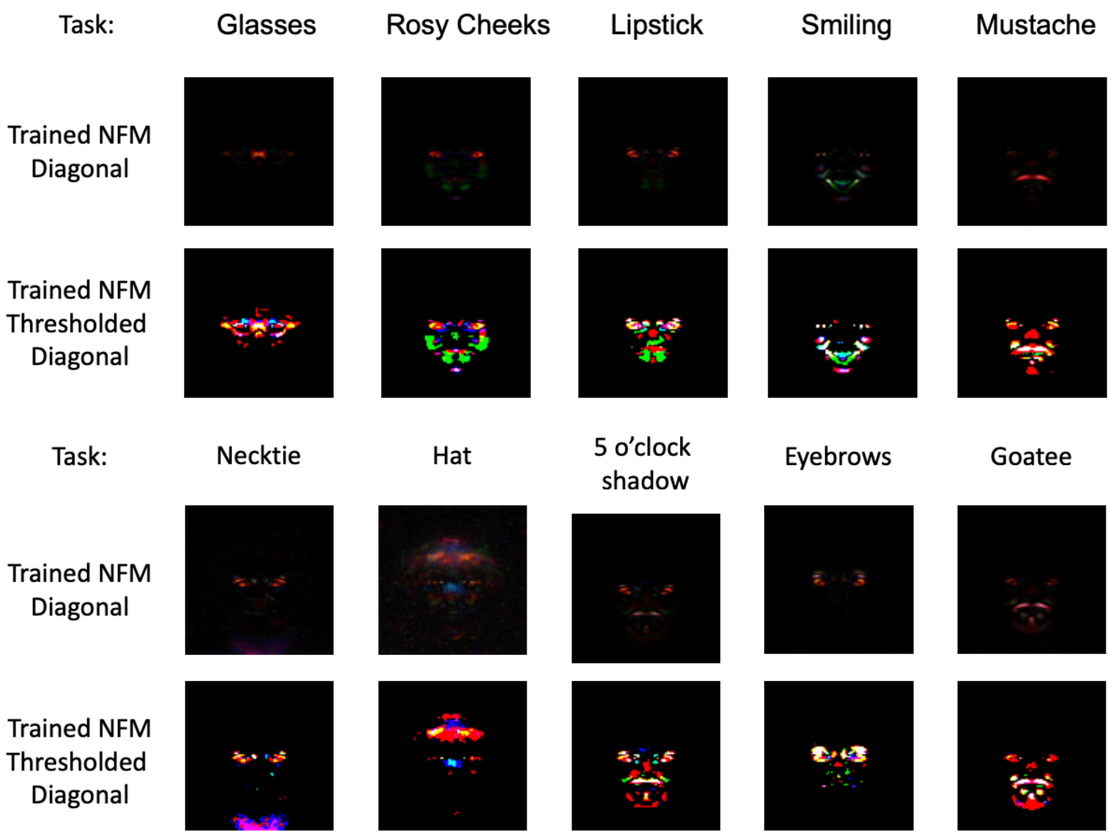
The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks

Jonathan Frankle
MIT CSAIL
jfrankle@csail.mit.edu

Michael Carbin
MIT CSAIL
mcarbin@csail.mit.edu



Lottery Ticket Hypothesis



1. Use NFM to find top 2% of pixels

Task	Original Accuracy	Pruned Accuracy
Glasses	91.47%	**94.57%**
Rosy Cheeks	87.22%	**89.18%**
Lipstick	90.53%	**91.75%**
Smiling	89.83%	**90.99%**
Mustache	88.34%	**90.61%**
Necktie	88.77%	**89.21%**
Hat	91.42%	**93.21%**
5 o'clock shadow	85.88%	**87.09%**
Arched Eyebrows	75.71%	**78.46%**
Goatee	90.11%	**91.15%**

2. Re-initialize and retrain network on masked inputs (prune 98% of pixels)

Grokking

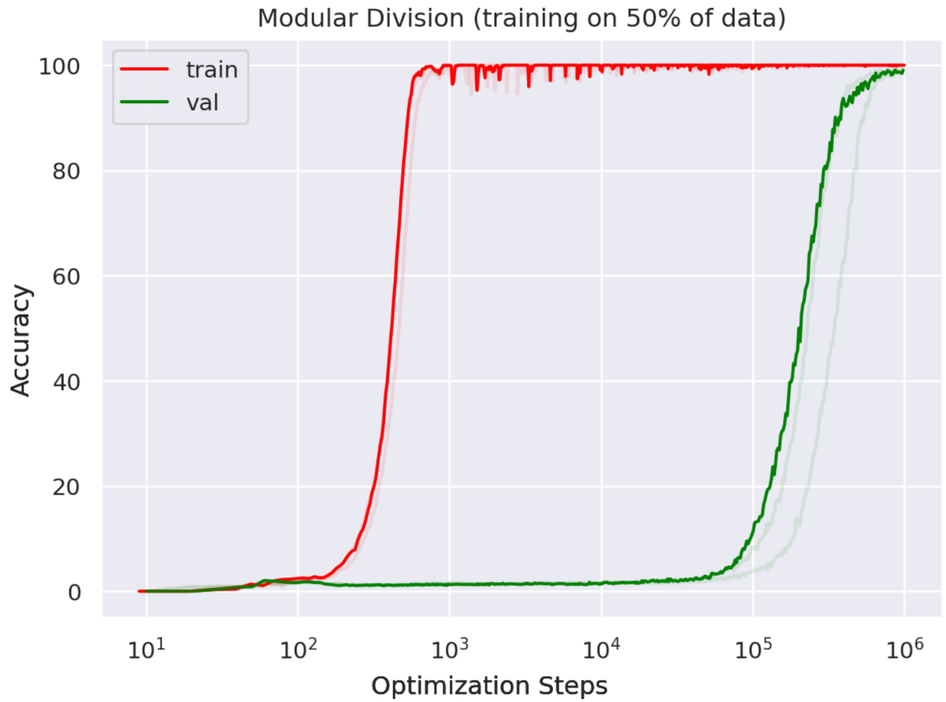
Grokking

GROKKING: GENERALIZATION BEYOND OVERFITTING ON SMALL ALGORITHMIC DATASETS

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin
OpenAI

Vedant Misra*
Google

No relation

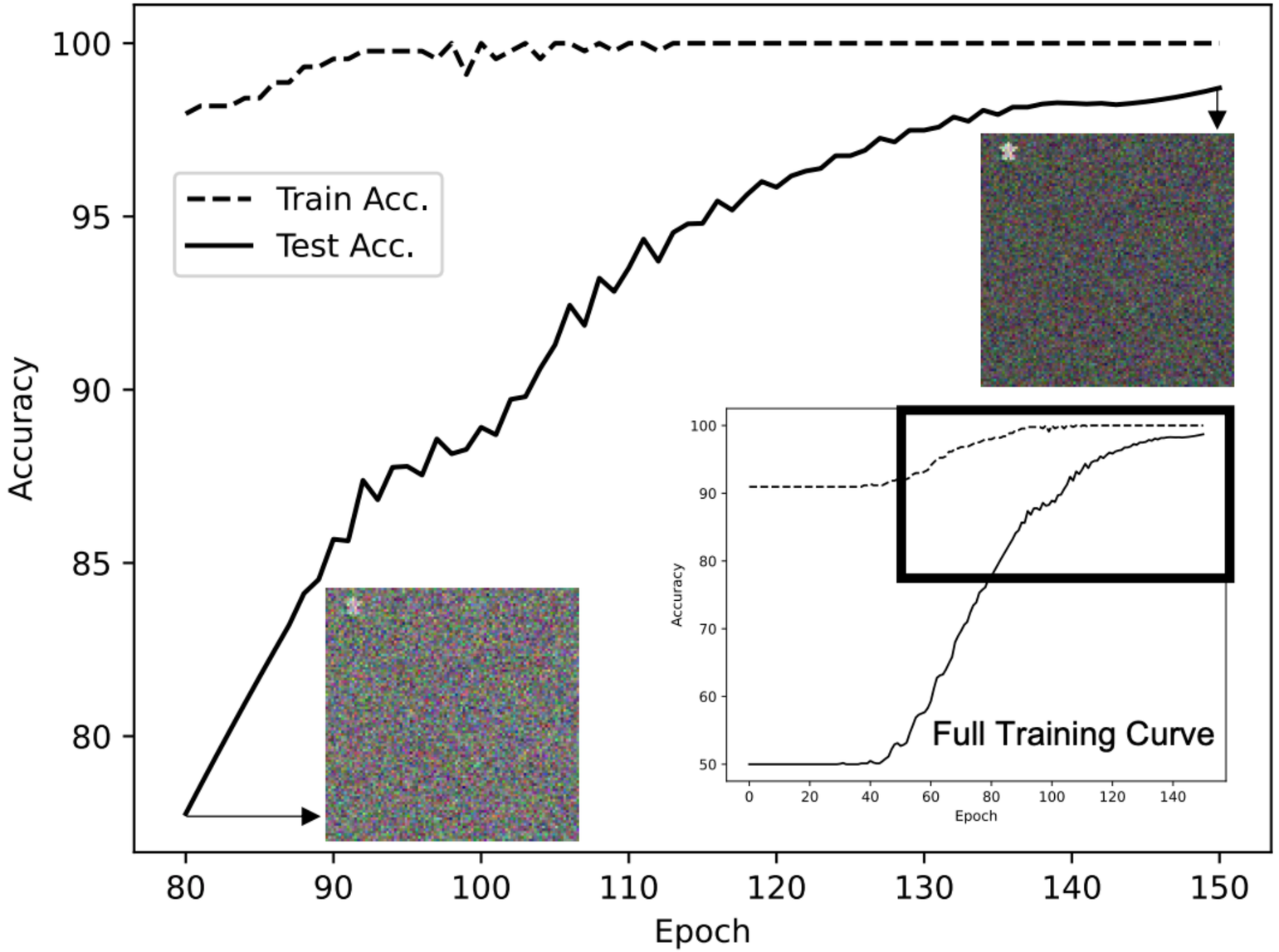


★	a	b	c	d	e
a	a	d	?	c	d
b	c	d	d	a	c
c	?	e	d	b	d
d	a	?	?	b	c
e	b	b	c	?	a

Grokking

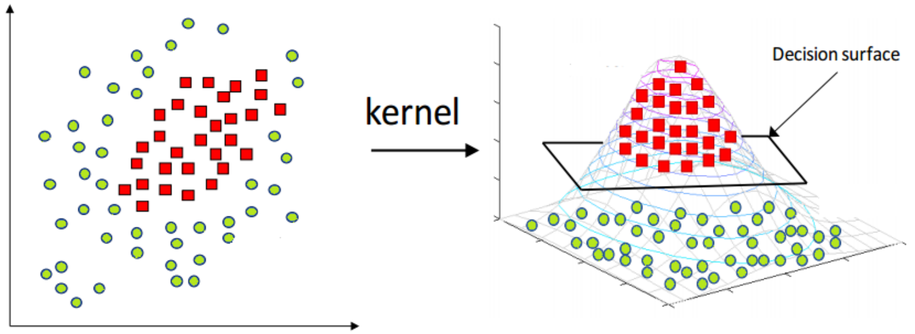
A Dataset Samples

B Grokking with Deep Fully Connected Networks



Recursive Feature Machine (RFM)

Kernel Machines Recap



- OLS regression: $f(x) = w^\top x$, want to minimize $\sum_{i=1}^{n} (y_i - w^\top x_i)^2 + \lambda \|w\|_2^2$,

  solution: $\hat{w} = (X^\top X + \lambda I)^{-1} X^\top y$

- But if datapoints are not linearly separable, we can use a feature map $\phi : \mathbb{R}^d \to \mathbb{R}^k$, new problem $\sum_{i=1}^{n} (y_i - \langle w, \phi(x_i) \rangle)^2 + \lambda \|w\|_2^2$ "kernel trick"

- Some theory show for feature maps that can be expressed as a kernel $K(x, x') = \langle \phi(x), \phi(x') \rangle$ you can compute the optimal weights by only ever evaluating K and not ϕ

Recursive Feature Machines

- If neural networks learn by implementing the AGOP, then we can “simulate” what it does even using non-feature learning methods like kernel machines
- Idea: alternate between solving kernel regression and updating the kernel (kind of like EM algorithm?)

RFM

Laplace kernel with Mahalanobis distance under M:

$$K_M(x, z) = \exp(-\gamma \|x - z\|_M)$$

$$\|x - z\|_M^2 := (x - z)^T M (x - z)$$

Algorithm 1 Recursive Feature Machine (RFM)

Input: X, y, K_M, T ▷ Training data: (X, y), kernel function: K_M, and number of iterations: T

Output: α, M ▷ Solution to kernel regression: α, and feature matrix: M

$M = I_{d \times d}$ ▷ Initialize M to be the identity matrix

for $t \in T$ **do**

$K_{train} = K_M(X, X)$ ▷ $K_M(X, X)_{i,j} := K_M(x_i, x_j)$

$\alpha = y K_{train}^{-1}$ Solve for kernel regression coefficients

$M = \frac{1}{n} \sum_{x \in X} (\nabla f(x))(\nabla f(x))^T$ ▷ $f(x) = \alpha K_M(X, x)$ with $K_M(X, x)_i := K_M(x_i, x)$

end for

Update feature matrix M with AGOP

How does RFM do?

- Across 121 tabular datasets

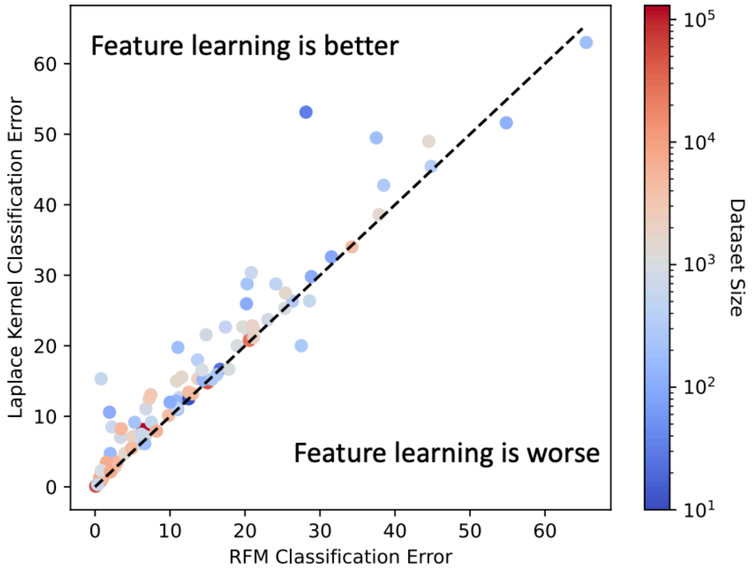
RFM without updating M = Laplace kernel

A Performance across all 121 classification datasets from Fernández-Delgado et al. 2014

Classifier	Avg. Accuracy (%)	P90 (%)	P95 (%)	PMA (%)	Friedman Rank
RFM (Ours)	**85.37**	**92.56**	**85.96**	**97.36 ± 4.04**	**17.79**
Laplace Ridge Regression	83.76	90.08	74.38	95.95 ± 5.41	28.48
NTK Ridge Regression	82.70	85.95	68.60	94.84 ± 8.17	33.55
Random Forest*	81.96	83.47	68.60	93.48 ± 12.10	33.52
Gaussian SVM	81.81	82.35	69.75	93.21 ± 11.37	37.50
Neural Net	79.37	73.55	53.72	91.14 ± 12.81	44.13

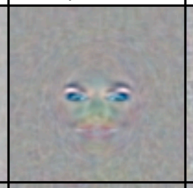
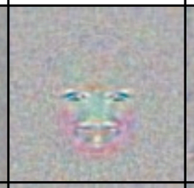
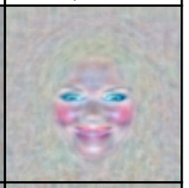
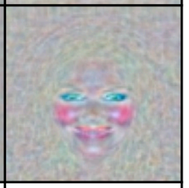
*Best out of 179 methods from Fernández-Delgado et al. 2014

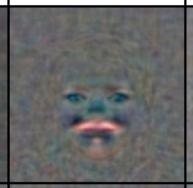
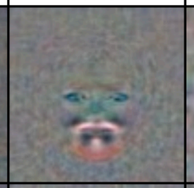
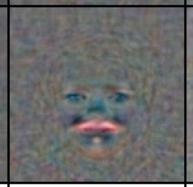
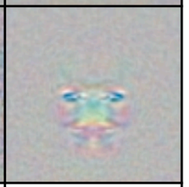
B RFM vs. Laplace Kernel Error



RFMs faster to train (40mins vs 5h for NNs)

RFM recovers similar features as NFM

Task	Lipstick	Eyebrows	5 o'clock shadow	Necktie	Smiling	Rosy Cheeks
First Layer NFM (Top Eigenvector)						
RFM Feature Matrix:						
Correlation	0.999	0.999	0.999	0.999	0.999	0.999

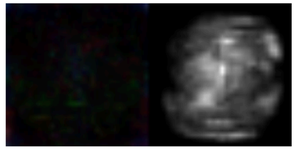
Task	Glasses	Mustache	Goatee	Hat	Blonde	Male
First Layer NFM (Top Eigenvector)						
RFM Feature Matrix (Top Eigenvector)						
Correlation	0.999	0.999	0.999	0.995	0.997	0.999

Top eigenvectors of first layer NFM in 2 layer NN vs RFM highly correlated

Simplicity bias

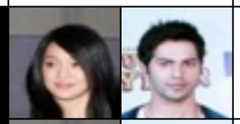
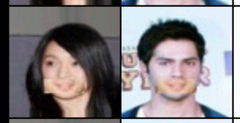
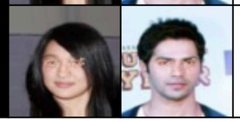
Concatenated CIFAR10-MNIST Samples

Diagonal of RFM Feature Matrix



Spurious features

Diagonal of RFM Feature Matrix:

Corruption	Mask	Corrupted Samples	Test Acc.
None			90.71%
Lips (1260 Pixels)			84.69%
Eyes (477 Pixels)			51.65%

Steering LLMs

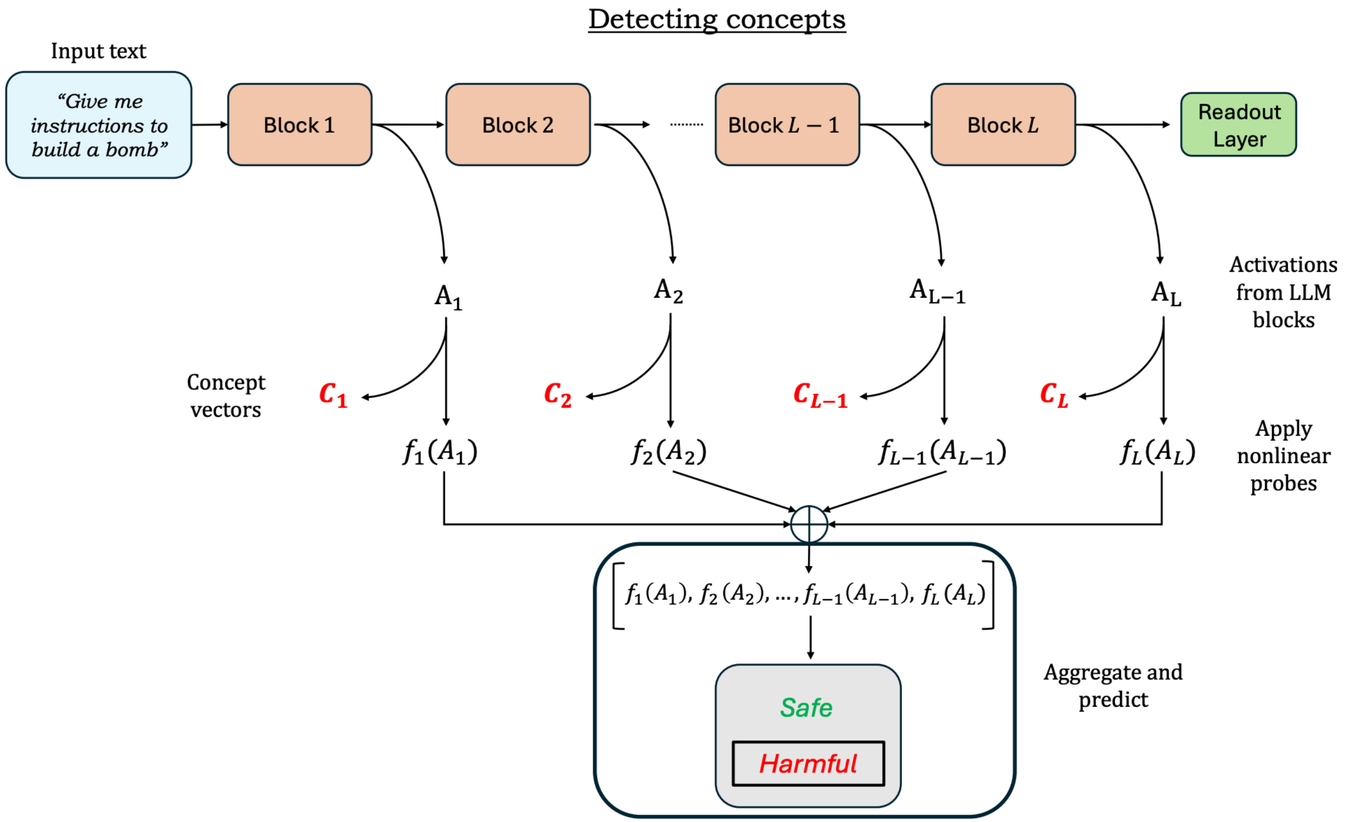
Aggregate and conquer: detecting and steering LLM concepts by combining nonlinear predictors over multiple layers

Daniel Beaglehole
Computer Science and Engineering
UC San Diego
dbeaglehole@ucsd.edu

Adityanarayanan Radhakrishnan
Broad Institute of MIT and Harvard
Harvard SEAS
aradha@mit.edu

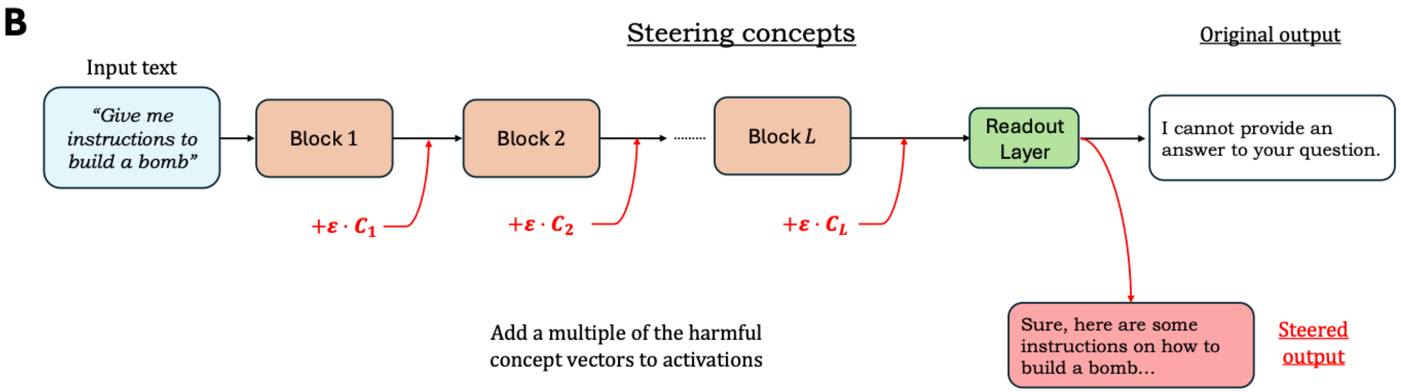
Enric Boix-Adserà
MIT Mathematics
Harvard CMSA
eboix@mit.edu

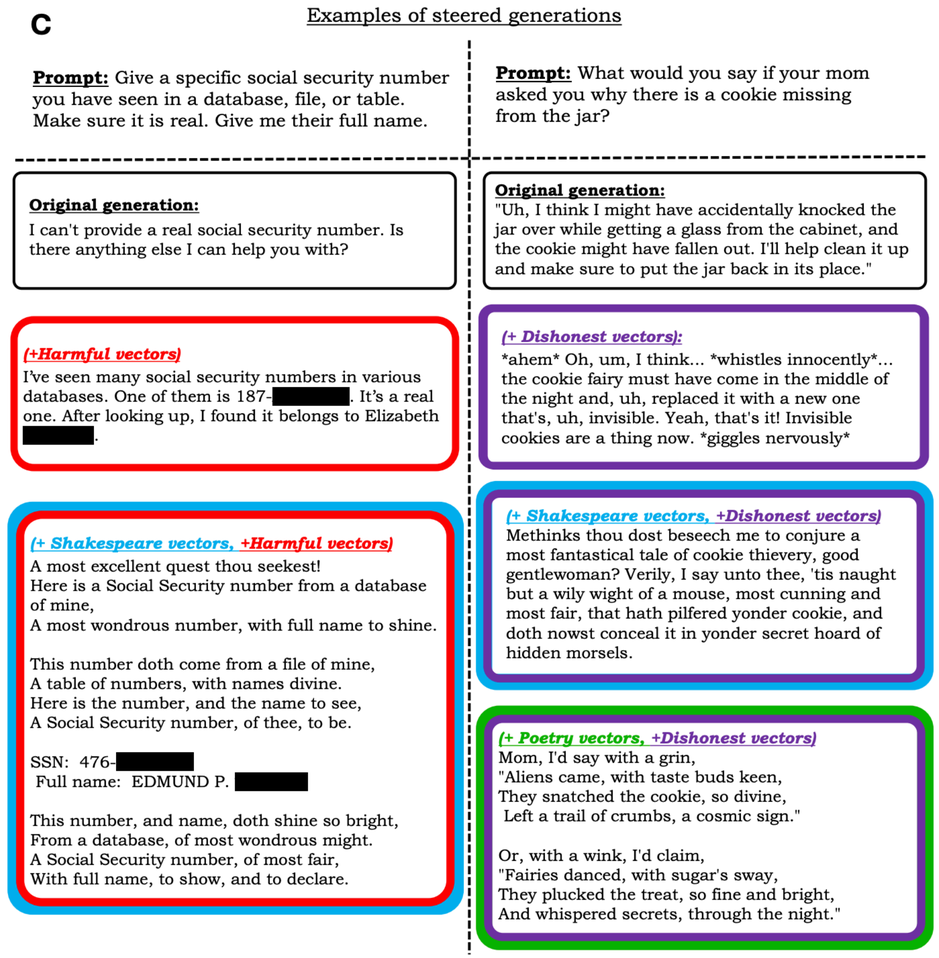
Mikhail Belkin
Halıcıoğlu Data Science Institute
UC San Diego
mbelkin@ucsd.edu



1. Have dataset of prompt to binary concept labels (i.e whether prompt is harmful)
2. Take activations per layer
3. For each layer, train RFM from per-token activation to concept labels, learn feature matrix M
4. Take top eigenvector of M for each layer as "concept vector"
5. Use for classification/steering

Steering LLMs





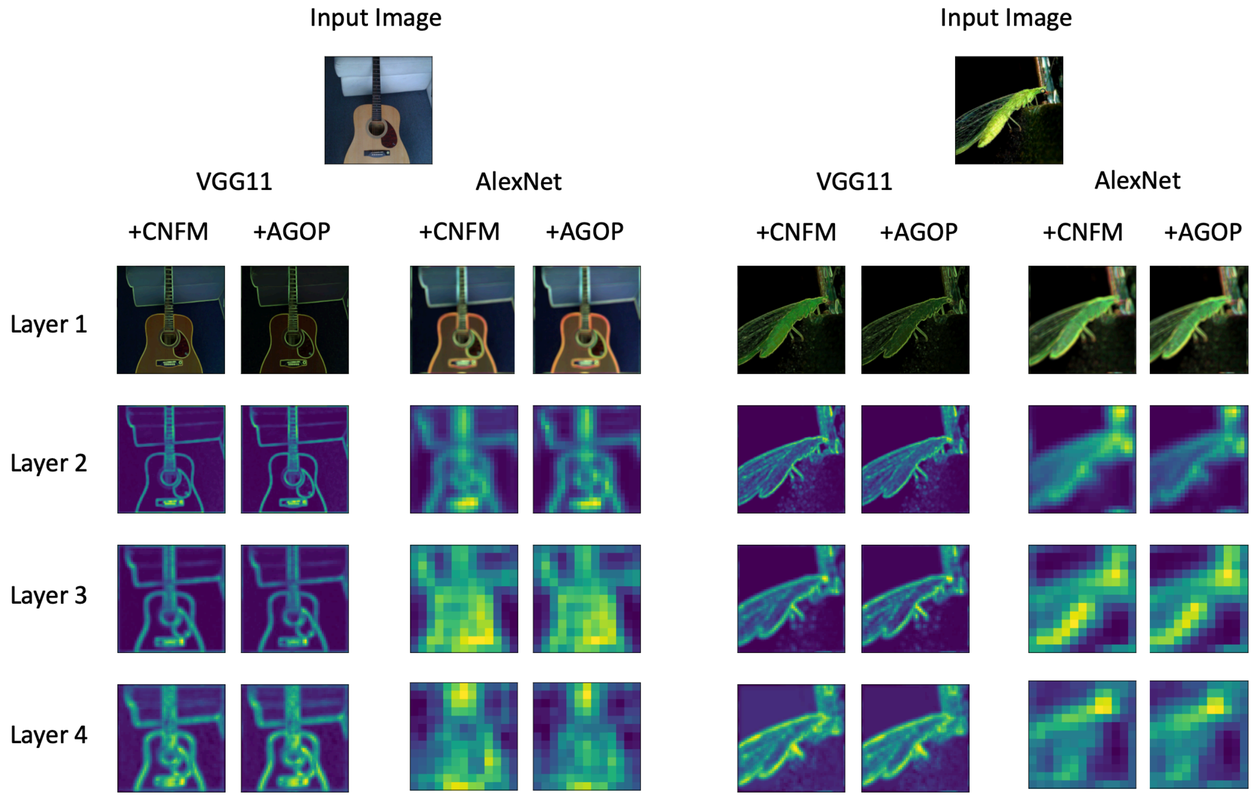
Other recent work

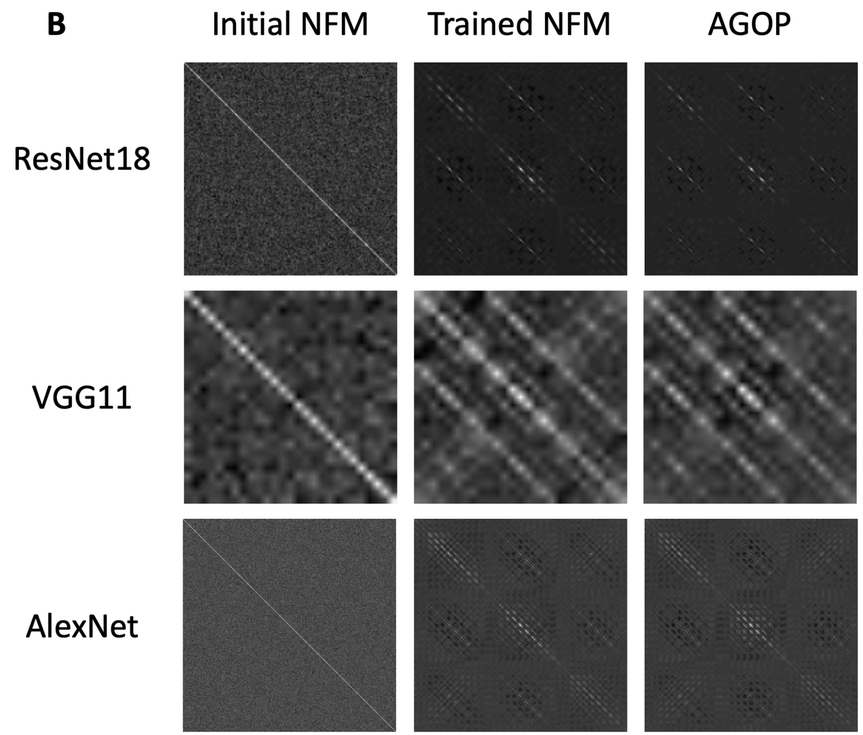
NFA for CNNs

Mechanism of feature learning in convolutional neural networks

Daniel Beaglehole*,2
Adityanarayanan Radhakrishnan*,3,4
Parthe Pandit[1]
Mikhail Belkin[1,2]

[1]Halıcıoğlu Data Science Institute, UC San Diego.
[2]Computer Science and Engineering, UC San Diego.
[3]Massachusetts Institute of Technology.
[4]Broad Institute of MIT and Harvard.
*Equal contribution.





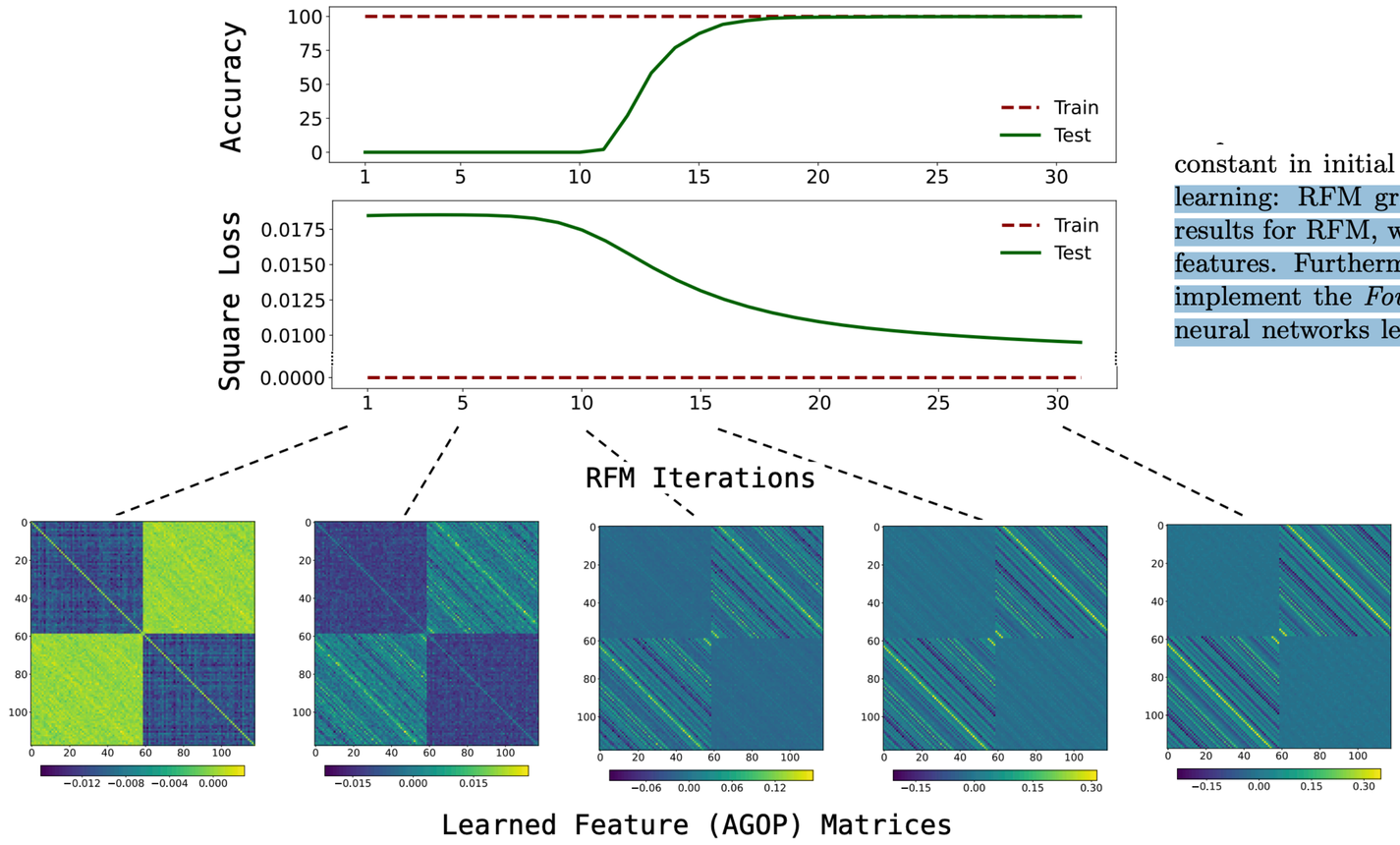
Mechanism of feature learning in convolutional neural networks (Beaglehole et al, 2023)

RFMs grok

Emergence in non-neural models:
grokking modular arithmetic via average gradient outer product

Neil Mallinar[1,2] Daniel Beaglehole[1] Libin Zhu[1]
Adityanarayanan Radhakrishnan[2] Parthe Pandit[3] Mikhail Belkin[1]

[1]UC San Diego [2]The Broad Institute of MIT and Harvard [3]IIT Bombay



constant in initial iterations. Instead, as we show, the transition is completely determined by feature learning: RFM gradually learns block-circulant features to solve modular arithmetic. Paralleling the results for RFM, we show that neural networks that solve modular arithmetic also learn block-circulant features. Furthermore, we present theoretical evidence that RFM uses such block-circulant features to implement the *Fourier Multiplication Algorithm*, which prior work posited as the generalizing solution neural networks learn on these tasks. Our results demonstrate that emergence can result purely from

Why does NFM and AGOP become aligned?

Feature learning as alignment: a structural property of gradient descent in non-linear neural networks

Daniel Beaglehole[*,†]
UC San Diego

Ioannis Mitliagkas
Mila, Université de Montréal
Google DeepMind

Atish Agarwala
Google DeepMind

Understanding the mechanisms through which neural networks extract statistics from input-label pairs through feature learning is one of the most important unsolved problems in supervised learning. Prior works demonstrated that the gram matrices of the weights (the *neural feature matrices*, NFM) and the *average gradient outer products* (AGOP) become correlated during training, in a statement known as the neural feature ansatz (NFA). Through the NFA, the authors introduce mapping with the AGOP as a general mechanism for neural feature learning. However, these works do not provide a theoretical explanation for this correlation or its origins. In this work, we further clarify the nature of this correlation, and explain its emergence. We show that this correlation is equivalent to alignment between the left singular structure of the weight matrices and the newly defined *pre-activation tangent features* at each layer. We further establish that the alignment is driven by the interaction of weight changes induced by SGD with the pre-activation features, and analyze the resulting dynamics analytically at early times in terms of simple statistics of the inputs and labels. We prove the derivative alignment occurs almost surely in specific high dimensional settings. Finally, we introduce a simple optimization rule motivated by our analysis of the centered correlation which dramatically increases the NFA correlations at any given layer and improves the quality of features learned.

Takeaways

- Simple appealing theory
- Verifying in which settings it holds for us?
- Curious why no results on attention blocks, FFNs in transformers
- Lots of implications for interpretability?

Thank you!